SIO4

4 Channel High-Speed
Serial Interface

Windows 98\NT\2K\XP Driver User Manual

Manual Revision: August 29, 2003

General Standards Corporation
8302A Whitesburg Drive
Huntsville, AL 35802
Phone: (256) 880-8787
Fax: (256) 880-8788
URL: http://www.generalstandards.com
E-mail: sales@generalstandards.com
E-mail: support@generalstandards.com

Preface

Additional copies of this manual or other literature may be obtained from:

General Standards Corporation
8302A Whitesburg Dr.
Huntsville, Alabama 35802
Phone: (256) 880-8787
FAX: (256) 880-8788
URL: http://www.generalstandards.com
E-mail: sales@generalstandards.com

General Standards Corporation makes no warranty of any kind with regard to this material, including, but not limited to, the implied warranties of merchantability and fitness for a particular purpose. Although extensive editing and reviews are performed before release to ECO control, **General Standards Corporation** assumes no responsibility for any errors that may exist in this document. No commitment is made to update or keep current the information contained in this document.

General Standards Corporation does not assume any liability arising out of the application or use of any product or circuit described herein, nor is any license conveyed under any patent rights or any rights of others.

General Standards Corporation assumes no responsibility for any consequences resulting from omissions or errors in this manual or from the use of information contained herein.

General Standards Corporation reserves the right to make any changes, without notice, to this product to improve reliability, performance, function, or design.

The Purchaser of this software may use or modify in source form the subject software, but not to re-market or distribute it to outside agencies or separate internal company divisions. The software, however, may be embedded in the Purchaser's distributed software. In the event the Purchaser's customers require the software source code, then they would have to purchase their own copy of the software.

General Standards Corporation makes no warranty of any kind with regard to this software, including, but not limited to, the implied warranties of merchantability and fitness for a particular purpose and makes this software available solely on an "as-is" basis. **General Standards Corporation** reserves the right to make changes in this software without reservation and without notification to its users.

GSC is a trademark of **General Standards Corporation**.

PLX and PLX Technology are trademarks of PLX Technology, Inc.

Table of Contents

1. Scope

The Purpose of this document is to describe how to interface with the SIO4 Windows Driver API developed by General Standards Corporation (GSC). This software provides the interface between the “Application Software” and the SIO4 board.

The SIO4 Driver API Software executes under control of the Windows Operating System. The SIO4 is implemented as a standard Windows driver API written in “C” programming language. The SIO4 Driver API Software is designed to operate on CPU boards containing x86 processors.

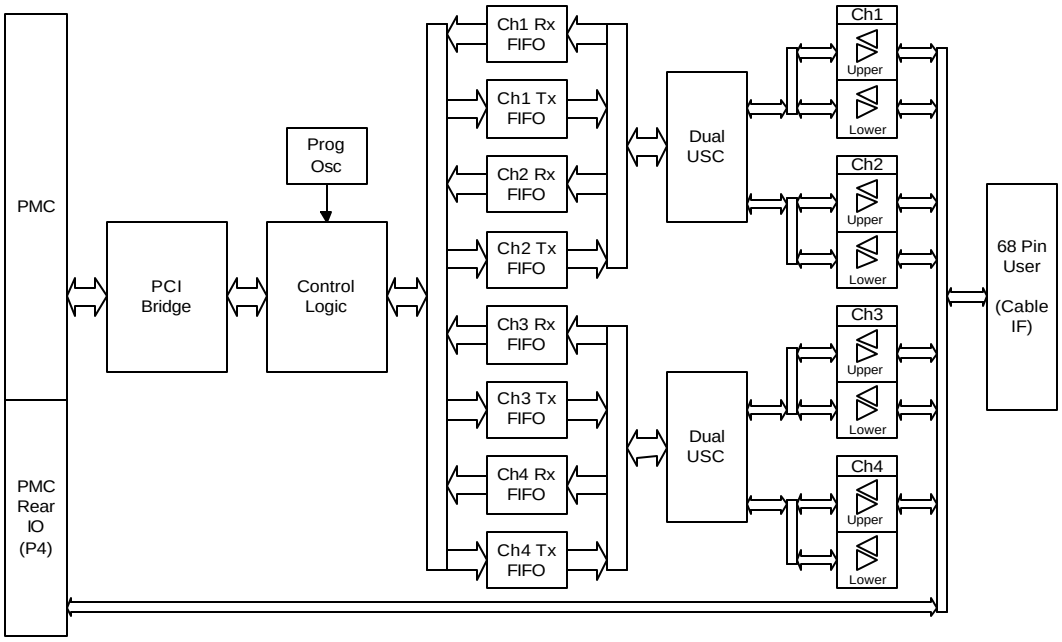
The SIO4 Driver consists of a Windows driver with an interface layer (GSC API) to simplify the interface to the PLX Driver. While an application may interface directly to the PLX driver, interfacing to the GSC API layer, will simplify the application software development.

2. Hardware Overview

The SI04 board is a four channel serial interface card which provides high speed, full-duplex, multi-protocol serial capability for PCI applications. The SIO4 combines two multi-protocol Dual Universal Serial Controllers (USC®) and 8 external FIFOs to provide four fully independent asynchronous or synchronous RS422/RS485 serial channels. These features, along with a high performance PCI interface engine, give the SIO4 unsurpassed performance in a serial interface card.

The SIO4 incorporates the following features:

- Four Independent Multi-Protocol Serial Channels
- Synchronous Serial Data Rates up to 10 Mbits/sec
- Asynchronous Serial Data Rates up to 1 Mbit/sec
- SCSI II type 68 pin front edge I/O Connector with optional cable adapter to four DB25 connectors.
- Independent Transmit and Receive FIFO Buffers for each Serial Channel – Up to 32k Deep Each
- Fast RS485/RS422 Differential Cable Transceivers to Provide Increased Noise Immunity
- Two Industry Standard Zilog Z16C30 Multi-Protocol Universal Serial Controllers (USC®)
- Serial Mode Protocols include Asynchronous, Bisync, SDLC, HDLC, Ethernet, and Nine-Bit
- Parity and CRC detection capability
- Two Serial Clocks, Two Serial Data signals, Data Carrier Detect, and Clear-To-Send per Channel
- Dual PCI DMA Engine to speed transfers and minimize host I/O overhead
- Four Programmable Oscillators provide increased flexibility for Baud Rate Clock generation



Sample Block Diagram of SIO4 (PMC-SIO4AR used as example)

3. Referenced Documents

The following documents provide reference material for the DMI32 board:

- PCI-SIO4 User's Manual – GSC
- PLX Technology, Inc. PCI 9080 PCI Bus Master Interface Chip data sheet.

4. General Standards API

This section describes the interface to the SIO4 GSC API. The SIO4 GSC API isolates the user from operating system specific requirements, allowing the API to be used with all Windows operating systems (98\NT\W2K\XP).

The SIO4 Win Driver provides an interface to a SIO4 card and a Windows application, which run on a x86 target processor. The driver is installed and devices are created when the driver is started during boot up. The functions of the driver can then be used to access the board. Devices are created with the name "board x" where "x" is the device number. Device numbers start at 1 and for each board found the device number will increment.

Included in the board driver software is a menu driven board application program. This program is delivered undocumented and unsupported but may be used to exercise the card and the device driver. It can also be used as an example for programming the SIO4 device.

The user interfaces to the GSC API at the basic level with the following functions:

- SIO4_FindBoards(): Detects all SIO4 Devices connected via the PCI Bus
- SIO4_Open(): Opens a driver interface to one SIO4 card.
- SIO4_Close(): Closes a driver interface to one SIO4 card.
- SIO4_Board_Reset(): Perform a reset on all Serial Channels of the board.
- SIO4_ReadLocalRegister(): Reads local registers from one SIO4 card.
- SIO4_WriteLocalRegister(): Writes to local Registers of one SIO4 card.
- SIO4_ReadUSCRegister(): Reads USC registers from one SIO4 card.
- SIO4_WriteUSCRegister(): Writes to USC Registers of one SIO4 card.
- SIO4_ReadConfigRegister(): Reads PCI Configuration registers from one SIO4 card.
- SIO4_WriteLocalRegister(): Writes PCI Configuration Registers of one SIO4 card.
- SIO4_EnableInterrupt(): Enables a desired interrupt.
- SIO4_DisableInterrupt(): Disables a desired interrupt.
- SIO4_Attach_Interrupt(): Attaches a user handle to a specified interrupt.
- SIO4_Dma_Mode(): Configures the DMA transfer mode (Demand Mode or Non-Demand Mode DMA operation)
- SIO4_Open_DMA_Channel(): Opens a DMA channel to transfer of data to or from the SIO4 – two DMA channels are available on the SIO4 cards.
- SIO4_DMA_From_FIFO(): Transfers data from the SIO4 channel FIFO to user memory.
- SIO4_DMA_To_FIFO(): Transfers data to the SIO4 channel FIFO from user memory.
- SIO4_Close_DMA_Channel(): Closes a DMA channel.

The user MUST call Find Boards to determine what PLX devices are installed in the system, and get the associated board number. The user then calls the Open function with each board number to be used. This function obtains a handle to the device and initializes the device parameters within the API / driver. The user is then free (assuming no errors) to write / read the registers as desired. The user should always call Close when done to free resources prior to exiting.

The function definitions and parameters are defined in the following paragraphs of this section.

4.1 SIO4_FindBoards()

Detects all PLX Devices connected via the PCI Bus.

Prototype:

```
U32  SIO4_FindBoards   (char      *pDeviceInfo,
                        U32       *ulError);
```

Returns – Total number of PLX boards found in the system or –1L if error or no boards found.

Where:

pDeviceInfo – Contains “Board #: Bus: Slot: Type: Ser#” info for PLX boards found.

ulError – Returns 0 or error code. Refer to tools.h for a list of error codes.

4.2 SIO4_Open

Initializes Handle for the passed board number IN THE DRIVER.

Prototype:

U32 SIO4_Open (**U32** BoardNumber);

Returns – Error code or 0.

Where:

BoardNumber – Defines board number to be used by the driver for a particular device.

Refer to tools.h for a list of error codes.

4.3 SIO4_ReadLocalRegister

Read a value from the board local register.

Prototype:

```
U32  SIO4_ReadLocalRegister  (U32     BoardNumber,
                              U32     ulRegister,
                              U32     *pulError);
```

Returns – Value Read from Register.

Where:

BoardNumber – Defines board number to be used by the driver for a particular device.

ulRegister – Local Register to read. Values defined in SIO4interface.h

pulError – Returns 0 or error code. Refer to tools.h for a list of error codes.

4.4 SIO4_WriteLocalRegister

Write a value to the board local register.

Prototype:

```
U32 SIO4_WriteLocalRegister (U32    BoardNumber,
                             U32    ulRegister,
                             U32    ulValue);
```

Returns – Error code or 0.

Where:

BoardNumber – Defines board number to be used by the driver for a particular device.

ulRegister – Register to write. Values defined in SIO4interface.h

ulValue – Value to write to the selected register.
Refer to the SIO4 user manual for all register / bit definitions.

4.5 SIO4_ReadUSCRegister

Read a value from the board USC register.

Prototype:

```
U32  SIO4_ReadUSCRegister  (U32     BoardNumber,
                            U32     ulChannel,
                            U32     ulRegister,
                            U32     *pulError);
```

Returns – Value Read from Register.

Where:

BoardNumber – Defines board number to be used by the driver for a particular device.

ulChannel – Serial Channel. Registers are channel specific.

ulRegister – USC Register to read. Values defined in SIO4interface.h

pulError – Returns 0 or error code. Refer to tools.h for a list of error codes.

4.6 SIO4_WriteUSCRegister

Write a value to the board USC register.

Prototype:

```
U32 SIO4_WriteUSCRegister (U32     BoardNumber,
                           U32     ulChannel,
                           U32     ulRegister,
                           U32     ulValue);
```

Returns – Error code or 0.

Where:

BoardNumber – Defines board number to be used by the driver for a particular device.

ulChannel – Serial Channel. Registers are channel specific.

ulRegister – Register to write. Values defined in SIO4interface.h

ulValue – Value to write to the selected register.
Refer to the SIO4 user manual for all register / bit definitions.

4.7 SIO4_ReadConfigRegister

Read a value from the board PCI Configuration register.

Prototype:

```
U32  SIO4_ReadConfigRegister  (U32     BoardNumber,
                               U32     ulRegister,
                               U32     *pulError);
```

Returns – Value Read from Register.

Where:

BoardNumber – Defines board number to be used by the driver for a particular device.

ulRegister – PCI Configuration Register to read. Refer to Users Manual.

pulError – Returns 0 or error code. Refer to tools.h for a list of error codes.

4.8 SIO4_WriteConfigRegister

Write a value to the board PCI Configuration register.

Prototype:

```
U32 SIO4_WriteConfigRegister (U32    BoardNumber,
                              U32    ulRegister,
                              U32    ulValue);
```

Returns – Error code or 0.

Where:

BoardNumber – Defines board number to be used by the driver for a particular device.

ulRegister – Register to write. Refer to Users Manual.

ulValue – Value to write to the selected register.
Refer to the SIO4 user manual for all register / bit definitions.

4.9 SIO4_Close

Closes the device handle and frees the resources.

Prototype:

U32 SIO4_Close (**U32** BoardNumber);

Returns – Error code or 0.

Where:

BoardNumber – Defines board number to be used by the driver for a particular device.

4.10 Interface Functions

These functions allow the user to perform certain operations on the board, without having to keep track of individual register values and bit definitions.

4.10.1 SIO4_Board_Reset

Perform a reset on all Serial Channels of the board.

Prototype:

U32 SIO4_Board_Reset (**U32** BoardNumber);

Returns – Error code or 0.

Where:

BoardNumber – Defines board number to be used by the driver for a particular device.

4.10.2 SIO4_Open_DMA_Channel

Opens a DMA channel for subsequent data transfers.

Prototype:

```
U32  SIO4_Open_DMA_Channel      (U32  BoardNumber
                                 U32  ulDMAChannel);
```

Returns – Error code or 0.

Where:

BoardNumber – Defines board number to be used by the driver for a particular device.

ulDMAChannel – DMA channel to open, valid for 0 or 1.

4.10.3 SIO4_DMA_From_FIFO

Transfers data from the SIO4 channel FIFO to user memory. DMA channel must have been previously opened.

Prototype:

```
U32  SIO4_DMA_From_FIFO    (U32     BoardNumber,
                            U32     ulSerChannel
                            U32     ulDMAChannel);
                            U32     ulWords);
                            U32*    uData);
```

Returns – Error code or 0.

Where:

BoardNumber – Defines board number to be used by the driver for a particular device.

ulSerChannel – Serial Channel. Valid for 1-4.

ulDMAChannel – DMA Channel. Valid for 0 or 1.

ulWords – Words to transfer.

uData – Pointer to user memory buffer.

4.10.4 SIO4_DMA_To_FIFO

Transfers data from user memory to the SIO4 channel FIFO. DMA channel must have been previously opened.

Prototype:

```
U32  SIO4_DMA_To_FIFO     (U32     BoardNumber,
                           U32     ulSerChannel
                           U32     ulDMAChannel);
                           U32     ulBytes);
                           U32*    uData);
```

Returns – Error code or 0.

Where:

BoardNumber – Defines board number to be used by the driver for a particular device.

ulSerChannel – Serial Channel. Valid for 1-4.

ulDMAChannel – DMA Channel. Valid for 0 or 1.

ulWords – Words to transfer.

uData – Pointer to user memory buffer.

4.10.5 SIO4_Close_DMA_Channel

Closes the DMA channel. Must have been previously opened or an error returned.

Prototype:

```
U32  SIO4_Close_DMA_Channel     (U32     BoardNumber,
                                 U32     ulDMAChannel);
```

Returns – Error code or 0.

Where:

BoardNumber – Defines board number to be used by the driver for a particular device.

ulDMAChannel – DMA Channel. Valid for 0 or 1.

4.10.6 SIO4_EnableInterrupt

Enables the desired interrupt in the local register, and for the PCI bus. See SIO4 User manual for interrupt sources. Unused interrupts should be disabled to maximize performance.

Prototype:

```
U32  SIO4_EnableInterrupt     (U32     BoardNumber,
                               U32     ulValue);
```

Returns – Error code or 0.

Where:

BoardNumber – Defines board number to be used by the driver for a particular device.

ulValue – The desired interrupt value to set, valid for 0 – 0xFFFF.

4.10.7 SIO4_DisableInterrupt

Disables specified interrupt(s) in the local register, and for the PCI bus.

Prototype:

```
U32  SIO4_DisableInterrupt    (U32    BoardNumber,
                               U32    ulValue);
```

Returns – Error code or 0.

Where:

BoardNumber – Defines board number to be used by the driver for a particular device.

ulValue – The desired interrupt value to clear, valid for 0 – 0xFFFF.

4.10.8 SIO4_Attach_Interrupt

Attaches a user handle to a specified interrupt.

Prototype:

```
U32  SIO4_Attach_Interrupt     (U32              BoardNumber,
                                HANDLE           *userHandle,
                                INTR_TYPE        ulInterrupt,
                                U32              ulLocalIntr);
```

Returns – Error code or 0.

Where:

BoardNumber – Defines board number to be used by the driver for a particular device.

userHandle – User supplied handle to attach to the interrupt.

ulInterrupt – Interrupt type to attach to, valid for Local, DmaCh0, DmaCh1.

ulLocalIntr – Local interrupt to attach to if ulInterrupt = Local.

5. Driver Installation

This section details driver installation on the target system. Any current driver previously installed for the SIO4 must be uninstalled prior to this installation to avoid interference.

To install the driver, API, and associated example files, insert the CD ROM into the drive and close the bay. The installation should commence automatically and display user prompts. Follow the onscreen instructions to complete the installation.

Should the installation fail to automatically start, Select ***Start*** ® ***Run*** ® ***Browse*** on the Windows toolbar/popup and browse to find ***Setup.exe*** on the CD ROM. Click on ***OK*** to commence the installation.

The following files are installed on the target system:

OS dependent\…\PciSIO4.sys
OS dependent\…\PlxApi.dll
Program Files\General Standards\SIO4 C\Example.exe
Program Files\General Standards\SIO4 C\SIO4 Driver C.dll
Program Files\General Standards\SIO4 C\SIO4 Driver C.lib
Program Files\General Standards\SIO4 C\SIO4intface.h
Program Files\General Standards\SIO4 C\SIO4 Example.c
Program Files\General Standards\SIO4 C\Tools.c
Program Files\General Standards\SIO4 C\Tools.h
Program Files\General Standards\SIO4 C\CioColor.h
Program Files\General Standards\SIO4 C\SIO4Cdriver.inf

6. Example Program

This section describes the example program, and the files required to develop an application.

The complied example program allows the user to exercise the installed device, while observing the I/O. To execute, double click on 'Example.exe'. Refer to the Driver Installation section for file location.

The source is provided to educate the user with the GSC API function calls and provide a working example to aid the user with application development. To build the example program using MS Visual C++, create a project and add the following files:

Source Files	® ***SIO4 Example.c***
	® ***Tools.c***
Header Files	® ***SIO4intface.h***
	® ***CioColor.h***
	® ***Tools.h***
Resource Files	® ***SIO4 Driver C.lib***

Select ***Build*** ® ***[ProjectName].exe*** on the toolbar.
NOTE: SIO4 Driver C.dll must be in the project directory to run the example.

Contact GSC for example programs (drivers) for other development environments (i.e LabVIEW™, LabWindows/CVI™, etc.)